

AUS9-2000-0499-US1

PATENT

DETECTION OF RESOURCE EXCEPTIONS

5

CROSS REFERENCE TO RELATED APPLICATIONS

The present invention is related to the following U.S. Patent Applications which are incorporated herein by reference:

Serial No. _____ (Attorney Docket No. AUS9-2000-0501-US1) entitled "Identifying Non-Externalized Text Strings That Are Not Hard-Coded" filed _____.

Serial No. _____ (Attorney Docket No. AUS9-2000-0498-US1) entitled "Pre-Processing Code to Identify and Modify Format of Keys" filed _____.

TECHNICAL FIELD

The present invention relates to the field of internationalization, and more particularly to a scanning program that detects resource exception errors prior to the run-time of the application.

BACKGROUND INFORMATION

Internationalization is a process of enabling a program, e.g., Java, to run internationally. That is, an internationalized program has the flexibility to run correctly in any country. An internationalized program must be able to read, write and manipulate localized text. Furthermore, an internationalized program must conform to local customs when

displaying dates and times, formatting numbers and sorting strings.

5 Internationalization is becoming increasingly important with the explosive growth of the Internet and the World Wide Web where an ever increasing number of computer users are from various locales. A locale represents a geographic, cultural or political region. One of the problems with internationalization involves the use of text strings that may be hard-coded in the program, e.g., Java. Hard-coded text strings refer to text that will not vary with the locale. That is, the text strings may appear in English even when the program is run on the French locale. Various object-oriented languages such as Java have developed tools to assist in developing internationalized programs and allowing text strings to appear in the language of the locale. A discussion of object-oriented programming languages and in particular Java is deemed appropriate.

10
15
20
25 In an object-oriented programming language such as Java, a class is a collection of data and methods that operate on that data. The data and methods taken together describe the state and behavior of what is commonly referred to as an object. An object in essence includes data and code where the code manipulates the data. Hence a software application may be written using an object-oriented programming language such as Java whereby the program's functionality is implemented using objects.

30 Unlike many programming languages, Java is compiled into machine independent code commonly referred to as bytecodes instead of machine dependent code, i.e. executable code. Bytecodes are stored in a particular file format commonly referred to as a "class file" that includes bytecodes for methods of a class. In addition to the bytecodes for methods

of a class, the class file includes a symbol file as well as other ancillary information.

5 A computer program embodied as Java bytecodes in one or more class files is platform independent. The computer program may be executed, unmodified, on any computer that is able to run an implementation of what is commonly referred to as a Java virtual machine. The Java virtual machine is not an actual hardware platform, but rather a low level software emulator that can be implemented on many different computer processor architectures and under many different operating systems. The Java virtual machine reads and interprets each bytecode so that the instructions may be executed by the native processor. Hence a Java bytecode is capable of functioning on any platform that has a Java virtual machine implementation available. However, bytecode interpretation detracts from processor performance since the microprocessor has to spend some of its processing time interpreting bytecode instructions. Compilers commonly referred to as "just in time (JIT)" were developed to improve the performance of Java virtual machines. A JIT compiler translates Java bytecodes into the processor's native machine code during runtime. The processor then executes the compiled native machine code.

10
15
20
25
30 As stated above Java has developed tools to assist in developing internationalized programs and allowing text strings to appear in the language of the locale. One such tool is the use of resource files commonly referred to in Java as resource bundles. A resource bundle class may be used for externalizing text strings, i.e. messages. By externalizing text strings, appropriate text strings appear in the language of the locale. The resource bundle class is an associative array of keys and values. Keys are free formatted strings that appear in the program code as well as in the

5 resource bundle thereby allowing the program to access the
externalized string. Externalized strings may be represented
as a value associated with the key. That is, appropriate text
strings for a given locale are indexed by keys. Hence, the
program may access an externalized string by accessing the
value associated with the key in the resource bundle, i.e. the
key that matches the key in the program code. By having
resource bundles associated with particular locales, e.g., a
resource file with resources associated with the US English
10 locale, a resource file with resources associated with the
French locale and so forth, appropriate text strings
associated with the particular locale may be loaded at
runtime.

15 Unfortunately, software developers may not define a
key-value pair in the resource file, e.g., resource bundle, or
mistype the key, i.e. free formatted string, in either the
program code or the resource file source code thereby
resulting in a resource exception error during the run-time of
the application, e.g., Java. For example, a key-value pair in
the resource file, e.g., resource bundle, may not be defined
20 when there is no key or value associated with the key in the
resource file, e.g., resource bundle. Furthermore, software
developers may mistype the key, i.e. free formatted string, in
either the program code or in the resource file source code so
that the key, i.e. free formatted string, in the program code
25 and in the resource file source code do not match and thereby
prevent the program from accessing the externalized string in
the resource file. When a procedure, commonly referred to as
a method, in Java attempts to load an external string from the
resource bundle and either the key-value pair is not defined
30 in the resource file or the key in either the program code or
resource file source code is mistyped, a resource exception

5

error results. It would therefore be desirable to detect resource exception errors prior to the run-time of the application and subsequently define the key-value pair in the resource file, e.g., resource bundle, or correct the mistyped key in either the program code or resource file source code so as to avoid resource exception errors.

Patent Application No. 09/049,999

SUMMARY

5 The problems outlined above may at least in part be
solved in some embodiments by a scanning program that detects
method invocations that are used to open resource files to
detect resource exception errors. After identifying a method
10 invocation, the scanning program scans for a method signature,
i.e. pointer, associated with the identified method
invocation. The method signature points to the resource file
to be opened by the method invocation. The scanning program
further scans for a pair of string delimiters adjacent to the
method signature. The string within the pair of string
15 delimiters is a key to the resource file to be opened by the
method invocation. Once the resource file is opened, the
scanning program determines whether the key and the value
associated with the key are defined in the resource file. If
the key or its associated value is not defined in the resource
file, then a resource exception error is detected.

20 In one embodiment, a method for detecting resource
exceptions comprises the step of scanning a code for a method
invocation used to open a resource file. The method further
comprises the step of identifying the method invocation. The
method further comprises the step of scanning the code for a
method signature where the method signature points to the
25 resource file to be opened by the method invocation. The
method further comprises the step of scanning the code for a
pair of string delimiters adjacent to the method signature
where a string within the pair of string delimiters is a key
of the resource file. The method further comprises the step
30 of opening the resource file using the method invocation
identified to detect resource exception errors. If the key or

the associated value is not defined in the resource file, then a resource exception error is detected.

5 In another embodiment of the present invention, the method comprises the step of generating a report where the report may comprise a listing of all resource errors detected upon completing the scanning the code.

10 The foregoing has outlined rather broadly the features and technical advantages of the present invention in order that the detailed description of the invention that follows may be better understood. Additional features and advantages of the invention will be described hereinafter which form the subject of the claims of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

5 A better understanding of the present invention can be
obtained when the following detailed description is considered
in conjunction with the following drawings, in which:

Figure 1 illustrates a data processing system configured
in accordance with the present invention;

Figure 2 is a flowchart of a method for detecting
resource exception errors; and

Figure 3 illustrates key-value pairs in resource files.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191

DETAILED DESCRIPTION

5 The present invention comprises a method, computer
program product and data processing system for detecting
resource exception errors. In one embodiment of the present
invention, a scanning program scans for a method invocation
used to open a resource file. After identifying a method
10 invocation, the scanning program scans for a method signature,
i.e. pointer, associated with the identified method
invocation. The method signature points to the resource file
to be opened by the method invocation. The scanning program
further scans for a pair of string delimiters adjacent to the
15 method signature. The string within the pair of string
delimiters is a key to the resource file to be opened by the
method invocation. Once the resource file is opened, the
scanning program determines whether the key and the value
associated with the key are defined in the resource file. If
20 the key or its associated value is not defined in the resource
file, then a resource exception error is detected. In another
embodiment of the present invention, a report may be generated
where the report may comprise a listing of all resource errors
detected upon completing the scanning the code. It is noted
25 that even though the following discusses the present invention
in conjunction with a Java programming environment the present
invention may be implemented in any type of programming
environment where the programming language has the capability
of externalizing text strings in resource files.

30 Figure 1 - Computer System

Figure 1 illustrates a typical hardware configuration of data processing system 13 which is representative of a hardware environment for practicing the present invention. Data processing system 13 has a central processing unit (CPU) 10, such as a conventional microprocessor, coupled to various other components by system bus 12. An operating system 40, e.g., DOS, OS/2™, runs on CPU 10 and provides control and coordinates the function of the various components of Figure 1. An object-oriented programming system, such as Java 42, runs in conjunction with operating system 40 and provides output calls to operating system 40 which implements the various functions to be performed by the application 42. Read only memory (ROM) 16 is coupled to system bus 12 and includes a basic input/output system ("BIOS") that controls certain basic functions of data processing system 13. Random access memory (RAM) 14, I/O adapter 18, and communications adapter 34 are also coupled to system bus 12. It should be noted that software components including operating system 40 and application 42 are loaded into RAM 14 which is the computer system's main memory. I/O adapter 18 may be a small computer system interface ("SCSI") adapter that communicates with disk units 20, e.g., disk drive, and tape drives 40. It is noted that the scanning program of the present invention that detects resource exception errors may reside in disk unit 20 or in application 42. Communications adapter 34 interconnects bus 12 with an outside network enabling data processing system 13 to communication with other such systems. Input/Output devices are also connected to system bus 12 via a user interface adapter 22 and a display adapter 36. Keyboard 24, trackball 28, mouse 26 and speaker 30 are all interconnected to bus 12 through user interface adapter 22. Event data may be input to the object-oriented programming system through any

of these devices. A display monitor 38 is connected to system bus 12 by display adapter 36. In this manner, a user is capable of inputting to system 13 through keyboard 24, trackball 28 or mouse 26 and receiving output from system 13 via display 38 or speaker 30.

Preferred implementations of the invention include implementations as a computer system programmed to execute the method or methods described herein, and as a computer program product. According to the computer system implementations, sets of instructions for executing the method or methods are resident in the random access memory 14 of one or more computer systems configured generally as described above. Until required by the computer system, the set of instructions may be stored as a computer program product in another computer memory, for example, in disk drive 20 (which may include a removable memory such as an optical disk or floppy disk for eventual use in disk drive 20). Furthermore, the computer program product can also be stored at another computer and transmitted when desired to the user's work station by a network or by an external network such as the Internet. One skilled in the art would appreciate that the physical storage of the sets of instructions physically changes the medium upon which it is stored so that the medium carries computer readable information. The change may be electrical, magnetic, chemical or some other physical change.

Figure 2 - Method For Detecting Resource Exception Errors

Figure 2 illustrates a flowchart of one embodiment of the present invention of a method 200 for detecting resource exception errors prior to the run-time of the application 42. As stated in the Background Information section, software

5 developers may not define a key-value pair in the resource
file, e.g., resource bundle, or mistype the key, i.e. free
formatted string, in either the program code or the resource
file source code thereby resulting in a resource exception
error during the run-time of the application, e.g., Java.
That is, when a procedure, commonly referred to as a method,
in Java attempts to load an external string from the resource
bundle and either the key-value pair is not defined in the
resource file or the key in either the program code or
10 resource file source code is mistyped, a resource exception
error results. It would therefore be desirable to detect
resource exception errors prior to the run-time of the
application and subsequently define the key-value pair in the
resource file, e.g., resource bundle, or correct the mistyped
15 key in either the program code or resource file source code so
as to avoid resource exception errors. Method 200 is a method
that detects resource exception errors prior to the run-time
of application 42.

20 In step 210, a scanning program scans the code of an
application program 42, e.g., Java, line by line for a method
invocation for opening a resource file, e.g., resource bundle.
A method may be defined as an object-oriented term for a
procedure or function. One particular type of method may be a
procedure that is used to open a resource file, e.g., resource
25 bundle. As stated in the Background Information section, a
resource bundle class is an associative array of keys and
values. Keys are free formatted strings that appear in the
program code as well as in the resource bundle thereby
allowing the program to access the externalized string.
30 Externalized strings may be represented as a value associated
with the key. The following code depicts a resource bundle

class definition written in Java code illustrating resources,
i.e. key-value pairs, in a resource bundle.

```
5      public class Res extends java.util.ListResourceBundle {  
        static final Object [][] contents = {  
          { "Holiday Title", "Christmas" },  
          { "Month Title", "December" };  
        public Object [][] getContents() {  
          return contents;  
        }  
      }  
10  }
```

15 In this code, "Holiday Title" and "Month Title" are the keys.
The values associated with those keys are the text strings
"Christmas" and "December", respectively.

20 An example of resource files, e.g., resource bundles,
comprising key value pairs is illustrated in Figure 3. Figure
3 illustrates a plurality of resource files 310A-C where each
resource file may be associated with a particular locale. For
example, resource file 310A may be associated with a U.S.
English locale. Resource file 310B may be associated with a
German locale. Resource file 310C may be associated with a
French locale. These resource files, e.g., 310A-C, may
25 reside in a Java ARchive (JAR) file where the JAR file may
reside in disk unit 20. Resource files 310A-C may
collectively or individually be referred to as resource files
310 or resource file 310, respectively. It is noted that any
number of resource files may reside in a JAR file.

30 Referring to Figure 3, resource file 310A illustrates the
key-value pairs for the above illustrated Java code in the
U.S. English locale. Resource file 310B associated with a

5 German locale may have the key values of "Weihnachten" and
"Dezember" associated with the keys "Holiday Title" and "Month
Title", respectively. Resource file 310C associated with a
French locale may have the key values of "Noel" and "decembre"
10 associated with the keys "Holiday Title" and "Month Title",
respectively. Hence, a resource file 310 may be defined for
each locale an application 42 supports. The application 42,
e.g., Java, then loads the correct resource file 310 according
to the specified locale. For example, if the application 42
were running in the German locale, then when the interpreter
translates the key "Holiday Title", resource file 310B is
loaded into memory and the associated key value "Weihnachten"
is outputted.

15 An example of code, e.g., Java, that includes a method
invocation to open a resource file, e.g., resource bundle, is
shown below.

```
rbCal=ResourceLoader.getBundle("com.tivoli.uif.Resources.Calen  
darResources");  
20 setTitle(ResourceLoader.getString(rbCal, "Holiday Title");
```

25 ResourceLoader is a subclass created from the abstract class
ResourceBundle in Java. In the second line, getString() is a
method of the subclass that is used to open the resource
bundle rbCal. The getString() method in this subclass has two
variables which will be discussed in greater detail below.
Hence in step 210, the scanning program scans the code for a
method invocation for opening a resource file, e.g., resource
bundle. It is noted that the getString() method is exemplary
30 only and that there are other method invocations for opening a
resource file, e.g., resource bundle.

In step 220, a determination is made as to whether a method invocation was identified in step 210. If there was not a method invocation identified, then the scanning program generates a report in step 290 as will be described in greater detail below.

If a method invocation was identified in step 220, then the scanning program in step 230 scans the code for a method signature, i.e. pointer, of the method invocation previously identified in step 210. In one embodiment, the method signature, i.e. pointer, may be the first parameter of the method invocation, e.g., getString() method of the subclass ResourceLoader. The method signature, i.e. pointer, points to a resource file, e.g., resource bundle. In the above example of Java code, the method signature is rbCal which points to the resource bundle rbCal that is defined in the first line of the above code. The path name of the resource file, e.g., resource bundle, may be defined as a parameter in a method called getBundle(). In the subclass ResourceLoader of the abstract class ResourceBundle, the getBundle() method comprises one parameter that may be used for specifying the path name to the resource bundle. For example, in the above example of Java code, the path name to the resource bundle rbCal is "com.tivoli.uif.Resources.CalendarResources." Hence, the method signature rbCal points to the resource bundle rbCal and the resource bundle rbCal is loaded into memory via the getBundle() method which uses the path name "com.tivoli.uif.Resources.CalendarResources."

In step 240, the scanning program scans the code for a pair of string delimiters adjacent to the method signature, i.e. pointer, identified in step 230. That is, the scanning program scans the code for a string variable associated with the method invocation identified in step 210. In one

embodiment, the string variable may be the second parameter of the invocation method, e.g., getString() method of the subclass ResourceLoader. The string variable is a key of the resource file, e.g., resource bundle, that is pointed to by the method signature identified in step 230. For example, in the above Java code, the string "Holiday Title" adjacent to the method signature rbCal is a key of the resource bundle rbCal.

In step 250, the scanning program opens the resource file, e.g., resource bundle, identified by the method signature in step 230 using the method invocation identified in step 210, e.g., getString(). In the above example Java code, the getString() method of the subclass ResourceLoader may be used to open the resource bundle rbCal which was identified by the method signature, i.e. pointer, of rbCal. The path name of the resource bundle rbCal to be opened may be defined as a parameter of the getBundle() method used to define the resource bundle rbCal.

In step 260, a determination is made by the scanning program as to whether the key string identified in step 240 as well as the value associated with that key is defined in the resource file, e.g., resource bundle, identified in step 250. A determination as to whether the key identified in step 240 is defined in the resource file, e.g., resource bundle, refers to determining whether the key identified in step 240 matches any of the key(s) (if any are defined) in the resource file, e.g., resource bundle, identified in step 250. If the resource file, e.g., resource bundle, does not comprise either the key identified in step 240 or its associated value, then the scanning program detects a resource exception error in step 270. In one embodiment, the scanning program stores

resource exception errors detected that may later be printed out in a report in step 290.

5 If the resource file, e.g., resource bundle, does comprise a key string identified in step 240 and the associated value, then a determination is made as to whether there is more code to scan in step 280. If there is more code to scan, then the scanning program continues to scan the code line by line for a method invocation for opening a resource file, e.g., resource bundle, in step 210. If there is no more code to scan, then the scanning report may generate a report in step 290. In one embodiment, the report generated may comprise a listing of all the resource exception errors detected. In another embodiment, the report may further comprise information as to the particular line(s) of code comprising keys to resource files, e.g., resource bundles, where the keys or values associated with the keys are not defined in the associated resource file, e.g., resource bundle. A software developer may then correct the mistyped keys in either the program code or resource file, e.g., resource bundle, source code or define the keys or values associated with the keys in the resource file, e.g., resource bundle, thereby avoiding a resource exception error during the run-time of the application, e.g., Java.

10
15
20
25 Upon detecting a resource exception error in step 270, a determination is made as to whether there is more code to scan in step 280. If there is more code to scan, then the scanning program continues to scan the code line by line for a method invocation for opening a resource file, e.g., resource bundle, in step 210. If there is no more code to scan, then the scanning report may generate a report in step 290. In one embodiment, the report generated may comprise a listing of all the resource exception errors detected. In another

embodiment, the report may further comprise information as to the particular line(s) of code comprising keys to resource files, e.g., resource bundles, where the keys or values associated with the keys are not defined in the associated resource file, e.g., resource bundle. A software developer may then correct the mistyped keys in either the program code or resource file, e.g., resource bundle, source code or define the keys or values associated with the keys in the resource file, e.g., resource bundle, thereby avoiding a resource exception error during the run-time of the application, e.g., Java.

It is noted that the scanning program may reside in disk unit 20 or application 42. It is further noted that the scanning program of the present invention may be implemented to detect resource exception errors in any type of programming language that has the capability of externalizing text strings in resource files.

Although the method, computer program product and data processing system of the present invention is described in connection with several embodiments, it is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents, as can be reasonably included within the spirit and scope of the invention as defined by the appended claims. It is noted that the headings are used only for organizational purposes and not meant to limit the scope of the description or claims.